FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Segurança da Informação

Leonardo Rodrigues Ribeiro

Resumo:

DESEMPENHO DE ALGORITMOS QUÂNTICOS E CLÁSSICOS EM TREINAMENTO DE MACHINE LEARNING SUPERVISIONADO

Americana, SP

2023

Perspectiva clássica de machine learning quântico

A limitação de aprendizagem de máquinas está chegando devido ao aumento da quantidade de dados computacionais, foi pensando então na computação quântica que apresenta uma solução para esse problema ao usar os fenômenos quânticos, tais como a interferência e o emaranhamento, alinhados à Aprendizagem de Máquina, onde o chamamos de *Quantum Machine Learning* (QML).

A computação quântica é o estudo de processamento e armazenamento de informação quânticas cuja unidade fundamental, chamada de qubit, é representada por $\psi = \alpha 0 e 0 + \alpha 1 e 1$, em que $\alpha 0, \alpha 1 \in \mathbb{C}, |\alpha 0 \mid 2 + |\alpha 1 \mid 2 = 1$ e a probabilidade de valer 0 ou 1 é dado por $\{|\alpha 0|^2, |\alpha 1|^2\}$.

O algoritmo de ML ideal é aquele que se ajusta bem aos dados de entrada (complexidade de amostra) e tenha um bom desempenho na previsão do comportamento (complexidade de tempo).

Por se tratar de análise de grandes quantidades de dados, a computação quântica pode ser útil ao Machine Learning devido a capacidade de sobreposição de dados. Ainda que não seja comprovado que a implementação de ML em computador quântico reduza a quantidade de dados necessário (complexidade da amostra), ao considerar um que um learner tenha acesso aos dados em sobreposição e um oraculo, a melhoria da eficiência se dá pelo número de consultas ao oráculo (complexidade do tempo).

Algo primordial para a quantum machine learning é a transformada do clássico para o quântico, a RAM quântica carrega dados de maneira eficiente, mas seus recursos para implementação real não são vantajosos, por conta disso alguns algoritmos em machine learning usam a QRAM.

Concluímos ressaltando um ponto importante no *Quantum Machine Learning* que é a análise de ruídos, que pode aliviar problemas de ajuste de modelo em *Machine Learning* na computação clássica, espera-se que possa aproveitar os ruídos na computação quântica com a mesma finalidade.

Machine Learning Quântico

Definimos a aceleração quântica pela complexidade de consulta e de porta, sendo que essa complexidade se trata de quantas vezes o algoritmo precisa consultar a fonte de informação , já a complexidade de porta se trata de quantas operações elementares ou portas lógicas são necessários para executar o programa. Uma parte dos sistemas de machine learning utilizam operações em matrizes em espaços vetoriais de alta dimensões, base da qual parte a computação quântica, pois os próprios qubits são representados em um vetor de duas dimensões e as operações são feitas através de matrizes de dimensão 2^n.

O machine learning reconhece os mesmos padrões estatístico em dados de entrada e nos resultados que produz, é de se esperar que computadores quânticos, por produzir dados que são difíceis de serem reconhecidos por computadores clássicos,

também possam reconhecer dados difíceis de serem reconhecidos por programas de Machine Learning clássico, mas mesmo no quesito de processamento a aceleração quântica seja perceptível, existem os chamados problema de entrada e problema de saída, que trazem uma desaceleração significativa devido à dificuldade de colocar dados clássicos em estados quânticos e em ler os dados em computadores clássicos depois de passarem por processadores quânticos, machine Learning envolve uma grande coleta de dados e o tempo para carregar esses dados em computadores quânticos é exponencial tendo como modo alternativo o uso de QRAM mas tem um alto custo financeiro, também temos outro problema que é a correção do erro quântico pois é preciso fazer ajustes nas portas para que a interferência seja minimizada e a eficiência otimizada.

Demonstração da vantagem quântica em Machine Learning

A grande maioria dos algoritmos quânticos usam oráculo que é uma espécie de caixa preta com a solução de um problema difícil de resolver para um computador clássico, sendo que a eficiência de um programa vem da quantidade de vezes que o algoritmo precisou utilizar esse oráculo, ou seja, a eficiência dos métodos de quantum machine learning é proporcional ao tamanho do conjunto de dados (n) e ao número de agrupamentos ou classes (k).

Algoritmos quânticos para machine learning

Em grande maioria, os algoritmos quânticos são baseados em três técnicas, sendo elas, estimativa de fase de amplitude e simulação hamiltoniana, além delas há duas classes de algoritmos que possibilitam a solução de problemas que exigem muitos recursos computacionais, sendo elas a transformada quântica de Fourier de Shor e o algoritmo de Grover que permite encontrar o mínimo e o máximo em uma matriz desordenada.

K-Medias e K-Medianas

k-media: Utiliza a pesquisa de Grover para calcular os centroides mais próximos

k-medianas: Calcula a mediana de cada agrupamento e redistribuir todos os pontos de dados, para garantir que todos tenham pelo menos um elemento.

Máquina de Vetor Suporte e Regressão Linear

A Máquina de Vetor Suporte é um exemplo de como os algoritmos de Grover e HLL podem oferecer aceleração quântica para Machine Learning que de maneira geral, é um algoritmo classificador supervisionado que divide os pontos de dados em classes diferentes.

Dado um conjunto de treinamento $D = \{(x1, y1), \dots, (xm, ym)\}, com\ \vec{x}j \in \mathbb{R}d$ e os rótulos $yj \in \{+1, -1\}$ o SVM deve fazer uma separação otimizada no hiperplano $\vec{w}.x + b$ entre duas classes de dados, já a regressão linear trata-se de outro método de Machine Learning dado uma coleção de pontos de dados $(xiyi)i = 1$

N onde $(xi1, \ldots ., xim) \in \mathbb{R}m$ que são vetores de dimensão M com entradas $yi \in \mathbb{R}$ o algoritmo de regressão linear assume que existe uma função que relaciona $f(x) = w\,Tx$ caracterizada pelos parâmetros $w = (w1, \ldots , wM)$.

Máquinas de Vetor Suporte (SVM)

É um método de Machine Learning fundamentado na Teoria da Aprendizagem Estatística, originalmente desenvolvida para classificação binária através da construção de hiperplano de decisão onde busca construir um classificador f que produza a menor quantidade de predições erradas possíveis durante o treinamento. Sua vantagem é a otimização poder ser feita com utilizando inversão de matrizes com o algoritmo quântico HHL. Além disso, a aceleração quântica mapeando o espaço de características da máquina de vetor suporte para um computador quântico.

Análise de Componentes Principais – PCA

É um método de machine Learning que é usado para redução em massa de dados sem perda de informações, essa técnica é usada para agrupamento de indivíduos e geração de índices. É necessário ressaltar a importância do algoritmo PCA para tornar a entrada de dados mais eficiente uma vez que reduz o conjunto de dados sem diminuir a informação, podendo ser usado para otimizar outros algoritmos para machine learning além de ser usado para contornar o problema de entrada e saída de dados em computadores quânticos.

Dados

A eficiência de algoritmos de Machine Learning Quântico depende da velocidade em que são carregados no computador quântico, pois tem diferença nas situações em que os dados já chegam em um estado quântico ou se será necessário convertê-los. Para uma melhor eficiência na entrada de dados, temos 3 alternativas, sendo elas a codificação digital ou base, onde os dados são convertidos de sua forma binaria para qubits, A segunda opção é conhecida como codificação analógica e os vetores de dados clássicos são codificados em amplitudes dos qubits e a terceira é a memória QRAM (*Quantum Randon Memory Access*) que permite a consulta de endereços em sobreposição e retorna o dado correspondente na enésima célula de memória.

Implementação (Classificadores)

O método para gerar o classificador no *scikit learn* é o Classificador de Vetor Suporte, esse método pertence a biblioteca Python SVM. O trecho de código abaixo exemplifica a construção do classificador clássico:

$$classificador\_classico = SVC()$$

Criamos o classificador usando recursos quânticos é preciso de duas etapas, sendo elas a criação de mapa de recursos quânticos com o método *ZZFeatureMap* e depois é gerado o kernel quântico com o método *QuantumKernel*.

$$mapa\_recursos = ZZFeatureMap()$$

$$kernel\_quantico = QuantumKernel()$$

Logo após, o kernel gerado pode ser passado para o método QSVC que irá criar o classificador usando os recursos do computador de quântico.

$$classificador\_quantico = QSVC()$$

Após gerar os classificadores, é usado o método *fit* do *scikit learn* para treiná-los (com o conjunto de treinamento) e na sequência, o método *predict* com o conjunto de teste (para verificar a eficiência do classificador).

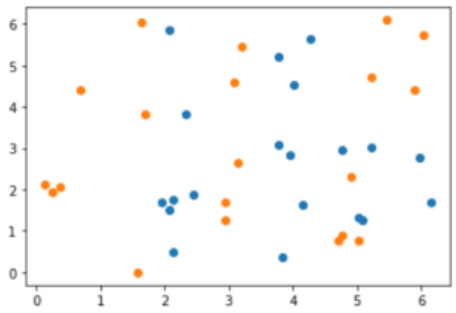
Treinamento de Dados

Abaixo é mostrado os processos de treinamento supervisionado com os conjuntos de dados AD HOC e DIGITS

Conjunto de dados AD HOC:

```
from qiskit_machine_learning.datasets import ad_hoc_data
adhoc_dimension = 2
train_features, train_labels, test_features, test_labels, adhoc_total = ad_hoc_data(
    training_size=20,
    test_size=5,
    n=adhoc_dimension,
    gap=0.3,
    plot_data=True, one_hot=False, include_sample_total=True
)
```

Importando o conjunto de dados é possível gerar o gráfico para facilitar a visualização do conjunto de dados e suas classes:



Construção do classificador com o método SVM e depois o treinamento do classificador com o método *fit*.

```
from sklearn import svm
classficador_sklearn_svm = svm.SVC(C=1.0)
classficador_sklearn_svm.fit(train_features,train_labels)
classficador_sklearn_svm.predict(test_features)
previsao = classficador_sklearn_svm.predict(test_features)
print(previsao)
```

Construção do kernel quântico:

```
adhoc_feature_map = ZZFeatureMap(feature_dimension=adhoc_dimension,
                                 reps=2, entanglement='linear')

adhoc_backend = QuantumInstance(BasicAer.get_backend('qasm_simulator'), shots=1024,
                                seed_simulator=seed, seed_transpiler=seed)

adhoc_kernel = QuantumKernel(feature_map=adhoc_feature_map, quantum_instance=adhoc_backend)
```

Testado o método clássico SVC no kernel quântico criado:

```
adhoc_svc = SVC(kernel=adhoc_kernel.evaluate)
adhoc_svc.fit(train_features, train_labels)
adhoc_svc.predict(test_features)
previsao_qkernel = adhoc_svc.predict(test_features)
print(previsao_qkernel)
```

Treinamento construindo-se um classificador com a função quântica qsvc(QISKIT):

```
qsvc = QSVC(quantum_kernel=adhoc_kernel)
qsvc.fit(train_features, train_labels)
qsvc.predict(test_features)
previsao_qsvc = qsvc.predict(test_features)
print(previsao_qsvc)
```

Foi utilizado o módulo métrico do *sklearn* para medir os resultados:

```
from sklearn.metrics import classification_report
target_names = ['class 0', 'class 1']
```

Treinamento do Conjunto de dados IRIS

Foi utilizado o mesmo modelo do conjunto de dados AD HOC, mas foi preciso usar a versão mais antiga da importação do conjunto de dados Iris para fazer duas alterações, sendo que a primeira alteração foi o formato dos dados dos rótulos, pois na versão atual do *Qiskit* os rótulos são dados em lista dentro do *array*, mas para passar para o método SVM do scikit learn era preciso que esses rótulos fossem do tipo inteiro. A segunda alteração diz respeito à própria classificação, uma vez que na computação quântica, dados os recursos atuais, a classificação binária é mais eficiente. Assim, ao invés de usar os três conjuntos de espécie da planta Iris, usamos somente 2 espécies do conjunto.

Importação do conjunto de treinamento Iris:

```
#alterei o nome da função para my_iris e one_hot para False
def my_iris(training_size, test_size, n, plot_data=False, one_hot=False):
    """returns iris dataset"""
    class_labels = [r"A", r"B"]
    data, target = datasets.load_iris(return_X_y=True)
    sample_train, sample_test, label_train, label_test = train_test_split(
        data, target, test_size=test_size, random_state=42
    )
```

Conclusão:

Com esse artigo pude ver que a aceleração quântica no campo de machine learning otimizará o processamento de dados e utilizando o conjunto de dados AD HOC, e que teremos uma vantagem quântica nos testes para taxas de acertos em relação ao Machine Learning supervisionado, sendo 100 % para a computação quântica e 70% para a computação clássica, já o conjunto de dados IRIS se mostrou mais vantagem tendo 100% de acuracidade em relação a computação quântica.

Entendi também que esses dados vêm em decorrer ao resultado de duas alterações realizadas no conjunto IRIS sendo o primeiro a relação ao formato dos dados dos rótulos e a segunda alteração, relacionada à própria classificação, uma vez que na computação quântica, dados os recursos atuais, a classificação binária é mais eficiente, utilizando dois ao invés de três conjuntos ao invés de dois e que teve a tentativa de introduzir um terceiro conjunto de dados, sendo ele o *Digits*, seguindo o mesmo modelo das implementações com o AD HOC e *IRIS*, mas não sendo possível gerar o kernel em computador quântico devido ao tamanho do conjunto de dados que possui imagens.

Para um futuro próximo, é de bom grado pesquisar um novo conjunto de dados e mais memória para computadores quânticos para que assim os dados sejam processados mais rápidos e a *Quantum Machine Learning* funcione com desempenho máximo e assim se implemente completamente no mundo e na sociedade no geral.